IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant:	Scott James Weaver	§	Group Art Unit:	2193
		§		
Serial No.:	10/041,743	§	Confirm. No.:	3989
		§		
Filed:	January 10, 2002	§	Examiner:	Tuan A. Vu
		§		
For:	DATA WEDGE	§	NCR Dkt. No.:	9288

BRIEF IN SUPPORT OF APPEAL

Sir:

This brief in support of appeal is filed in support of Applicant's notice of appeal filed on August 7, 2009, in response to a final rejection dated April 7, 2009, in this matter. In addition, Applicant filed a pre-appeal brief with the notice of appeal. On September 15, 2009, the Office issued a Notice of Panel Decision stating the case should proceed to the Board of Patent Appeals and Interferences. Please provide any extensions of time that may be necessary and charge any fees that may be due to Account No. 14-0225, but not to include any payment of issue fees.

(1) REAL PARTY IN INTEREST

The real party in interest in this matter is NCR Corporation, Dayton, Ohio, by virtue of an assignment recorded at reel 012473, frame 0122-0024, on January 10, 2002.

(2) RELATED APPEALS AND INTERFERENCES

Applicant is not aware of any appeals or interferences related to this patent application (serial no. 10/041,743).

(3) STATUS OF CLAIMS

Claims 1-16 have been cancelled.

Claims 17-33 are currently pending in the application and have been rejected two or more times. The rejection of these claims is being appealed. The claims, as currently pending, are listed in an Appendix to this Appeal Brief.

(4) STATUS OF AMENDMENTS

Applicant has filed no amendments after receipt of the April 7, 2009 final Office Action.

(5) SUMMARY OF CLAIMED SUBJECT MATTER

Note: Items in figures are referenced as Fig. #:item number (e.g. Fig. 1:12 for Figure 1, item 12).

(A) EXPLANATION:

Claimed are systems and a method for storing and translating data (Paragraphs 20-21) between a format of a first data model (Fig. 1:22) of a first software component (Fig. 1:12) and a format of a second data model (Fig. 1:24) of a second software component (Fig. 1:14) using a data wedge (Fig. 1:10) comprising a third data model (Fig. 2:30) and a data storage (Fig. 1:16).

(B) CLAIMS WITH SPECIFICATION SUPPORT

Claim	**Specification Support**
17. A computer-implemented method of storing and translating data between a format of a first data model of a first software component and a format of a second data model of a second software component, the method comprising:	Page 12, paragraphs 44-47 Page 5, paragraphs 20-23
creating a first schema comprising the first data model of the first software component;	Page 10, paragraph 37, Fig. 3:30; Page 6, paragraph 23, Fig. 1:22.
creating a second schema comprising the second data model of the second software component;	Page 6, paragraph 23, Fig. 1:24.
creating a third data model and a data storage in a data wedge by integrating the first schema and the second schema into the data wedge;	Page 10, paragraph 38, Fig. 3:32; Page 6, paragraph 22, Fig. 1:16.
receiving a data element in the format of the first data model of the first software component, translating the data element from the format of the first data model of the first software component to the format of the third data model in the data wedge and storing the translated data element in the data storage, by the data wedge; and	Page 10, paragraph 40, Fig. 3:34.
retrieving the data element from the data storage and translating the data element into the format of the second data model of the second software component by the data wedge after receiving a request for the data element from the second software component.	Page 11, paragraphs 41 and 43; Fig. 1:16.
25. A computer system for translating data	Page 12, paragraph 45, Fig.

between a format of a first data model of a first software component and a format of a second data model of a second software component, the system comprising:	3:300.
a processor; and	Page 12, paragraph 45, Fig. 3:304.
a memory coupled to said processor, the memory having stored therein data and sequences of instructions which, when executed by said processor, cause said processor to:	Page 12, paragraph 45, Fig. 3:306.
create a first schema comprising the first data model of the first software component;	Page 10, paragraph 37, Fig. 3:30; Page 6, paragraph 23, Fig. 1:22.
create a second schema comprising the second data model of the second software component;	Page 6, paragraph 23, Fig. 1:24.
create a third data model and a data storage in a data wedge by integrating the first schema and the second schema into the data wedge;	Page 10, paragraph 38, Fig. 3:32; Page 6, paragraph 22, Fig. 1:16.
receive a data element in the format of the first data model of the first software component, translate the data element from the format of the first data model of the first software component to a format of the third data model in the data wedge and store the translated data element in the data storage; and	Page 10, paragraph 40, Fig. 3:34.
retrieve the data element from the data storage and translate the data element into the format of the second data model of the second software component after receiving a request for the data element from the second software component.	Page 11, paragraphs 41 and 43; Fig. 1:16.
32. A computer system for translating data between a format of a data model of a first	Page 12, paragraph 45, Fig. 3:300.

software component and a format of a data model of a second software component, the system comprising:	
a processor; and	Page 12, paragraph 45, Fig. 3:304.
a memory coupled to said processor,	Page 12, paragraph 45, Fig. 3:306.
wherein said processor is configured to execute a sequence of instructions contained in said memory, the instructions comprising a data wedge including	Page 12-13, paragraph 47;
a first schema of the first software component and	Page 6, paragraph 23, Fig. 1:22.
a second schema of the second software component,	Page 6, paragraph 23, Fig. 1:24.
the data wedge configured to translate a data element from the format of the data model of the first software component in accordance with the first schema to a data model of the data wedge and	Page 10, paragraph 40, Fig. 3:34.
when a request is received from the second software component, translate the data element from the format of the data model of the data wedge to the format of the data model of the second software component in accordance with the second schema.	Page 11, paragraphs 41 and 43; Fig. 1:16.

(6) GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

(A) 112 Rejection of Independent Claims 17 and 25

Claims 17 and 25 stand rejected under 35 U.S.C. §112, first paragraph, as failing to comply with the written description requirement.

(B) 102(e) Rejection of Independent Claims 17, 25 and 32

Claims 17, 25 and 32 stand rejected under 35 U.S.C. §102(e) as being anticipated by Worden (U.S. Pub. No. 2003/0149934).

(C) 103(a) Rejection of Dependent Claims

Dependent claims 18, 20, 26, 27 and 33 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Worden.

(7) ARGUMENT

All rejected claims should be allowed over the cited prior art references for the reasons set forth below.

(A) 112(P1) Rejection of Independent Claims 17 and 25

The Office asserts that claims 17 and 25 contain subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor, at the time the application was filed, had possession of the claimed invention. (See Office Action mailed April 7, 2009 [hereafter "Final"], page 2, section 3, first paragraph.) Applicant does not agree with this rejection and respectfully traverses the Office's analysis and the rejections based thereon.

In re Count 1 (Final, section 3):

Applicant requires "creating a third data model and a data storage in a data wedge by integrating the first schema and the second schema into the data wedge."

The Office asserts "it is not recognized from the Specifications that a act of integrating 2 schemas takes place to yield a third model as recited." (Final, page 2,

section 3, paragraph 2.) The Office further states "the Wedge view (or third model) is NOT one with integrating/merging result from respective schema elements of at least two users." (Final, page 3, lines 14-15.) Applicant disagrees with these assertions.

The specification teaches "The Data Wedge attempts to minimize code modification when integrating software components by utilizing a data model of cooperating components instead of functional interfaces." (Specification, paragraph 22.) This passage specifically teaches "integrating software components" and to do so by utilizing "a data model" of the cooperating components. The specification clearly provides support for integrating multiple software components utilizing one data model, which contains information from the multiple cooperating software components. This data model is Applicant's third data model.

The specification teaches that a "client" is a software component, which reuses or accesses the functionality of a server software component. (See Specification, paragraph 3.) The specification further teaches "a client that uses the Wedge must first create a schema describing its logical data model … once the schema is defined, the client creates an instance of the Wedge in step 32 by providing its schema … If the Wedge already exists … the schema is dynamically integrated into that Wedge …." (Specification, paragraphs 37 and 38.) The specification teaches that each client, also known as a software component, that is using the Wedge first creates a schema describing its own logical data model. In addition, the specification teaches that each "schema is dynamically integrated into the Wedge." The specification also teaches "Each client component must define its logical view of the data (its data model) and provide this definition to the Wedge upon connection. The definition is in the form of a superset of the XML schema definition …." (Specification, paragraph 24, first and second sentence.) The specification clearly teaches that each software component has a data model and that upon connection to the Wedge a schema of the data model is provided to the Wedge to be dynamically integrated into the Wedge.

From these passages, it is clear that Applicant's specification does indeed contain the requisite teachings to reasonably convey to one skilled in the art that Applicant indeed had possession of the claimed inventions at the time the present application was filed.

The Office's assertions are therefore improper and Applicant requests that the rejection be withdrawn.

In re Count 2 (Final, section 4):

Applicant requires "creating a first schema comprising the first data model of the first software component; creating a second schema comprising the second data model of the second software component."

The Office asserts "creating a first schema …; creating a second schema …'. According to the Specifications, the data Wedge receives schemas from users (see schema A, B – Fig. 1) and there is not a description therein that explicitly mentions about (the method/system being claimed as) a software capability to create the schema A or B as mentioned above." (Office Action, page 3, section 4, first paragraph.) Applicant disagrees with this assertion.

Applicant disagrees with this assertion and specifically traverses the statement that "According to the Specifications, the data Wedge receives schemas from users (see schema A, B – Fig. 1)." The elements the Office has identified in Fig. 1 do not show the presences of a user or that the Data Wedge receives schemas from users. To the contrary, the Specification teaches "each client component must define its logical view of the data (its data model) and provide this definition to the Wedge upon connection. This definition is in the form of a … schema definition." (Specification, paragraph 24.) This passage teaches that each client component provides schema to the Data Wedge upon connection. The client component is not a person but a software component. In addition and as shown above, the specification teaches that clients are software components. Asserting that Applicant's "client" is a person or user is inconsistent with the teachings of the specification.

In a rebuttal argument, the Office asserts "The client here is understood [to be] a[n] online user that operates a instance of this Wedge tool …." (Final Office Action, page 12, lines 20-21.) This is simply not a correct assertion by the Office and is inconsistent with the teachings of the specification. The Office is referencing step 30 in Fig. 2, which states "CREATE SCHEMA." For this step the specification teaches, "In step 30, a client that uses the Wedge must first create a schema describing its logical data

model." (Specification, paragraph 37.) The Office asserts that the "client" is "a online user." This is an improper assertion. The specification in the paragraph 36 states "A process flow of a software component or client, e.g., component A12, using the Wedge 10 is described with reference to Figure 2." The specification distinctly teaches that Figure 2 is a process flow of a software component or client using the Wedge. There is nothing that would lead a person of ordinary skill in the art to conclude that this passage could mean an online user as asserted by the Office. It is clear that a client is equal to a software component. As shown above, the specification teaches a "client" is a software component. Nowhere in the specification is there a teaching that a person or online user performs these functions. To assert that a client is a person is not consistent with the teachings of the specification. The assertion by the Office is improper and Applicant requests that it be withdrawn.

From these passages, it is clear that Applicant's specification does indeed contain the requisite teachings to reasonably convey to an ordinary person skilled in the art that Applicant indeed had possession of the claimed inventions at the time the present application was filed. The Office's rejection is therefore improper and Applicant requests the Board of Appeals reverse the rejection.

(B) 102(b) Rejection of Independent Claims 17, 25 and 32

Applicant requires "creating a third data model and a data storage in a data wedge by integrating the first schema and the second schema into the data wedge." The "first schema" is required to be a data model of a first software component and the "second schema" is required to be a data model of a second software component. The Office asserts that Worden teaches these requirements and states "see Fig. 9 – Note: using Xmulator to map model to XML for 2 intended languages reads on first and second integration of respective model schema into the wedge or third model." Applicant disagrees with this assertion.

Worden discloses a method to translate a document written in one XML language to a document written in a second XML language. Worden teaches that the Xmulator uses an XSLT file to translate an input file, in one XML language, to an output file, in another XML language. The XSLT file contains syntactic and meaning information for

each language. (See Worden, paragraphs 357 – 363.) The Office asserts that syntactic and meaning information for each language is equivalent to Applicant's first and second schemas. This is simply not true. Applicant's schema must be for data models for software components. Data models for software components are not the same as syntactic and meaning information for a language. A language is not a software component. The Office as failed to establish that Worden shows or suggest at least these required elements.

Applicant requires "storing the translated data element in the data storage, by the data wedge." The data element here has been translated into the format of the third data model. The Office asserts that Worden teaches this requirement, however Applicant disagrees. The passages cited by the Office teach an intermediate file (para. 38) where Worden stores mappings information in the file and not a data element that has been translated as required by Applicant. Therefore, the Office as failed to establish that Worden shows or suggest at least these required elements.

The Office has failed to show or suggest that Worden discloses at least these elements of Applicant's claimed invention. Therefore, the rejection of independent claims 17, 25 and 32 is improper and Applicant requests that the Board of Appeals reverse the rejection.

Furthermore, the Court of Appeals for the Federal Circuit has stated, "To anticipate a claim, a reference must disclose every element of the challenged claim and enable one skilled in the art to make the anticipating subject matter." (Elan Pharms., Inc. v. Mayo Found., 346 F.3d 1051, 2003.) Applicant asserts that Worden's enablement is directed to a device designed translate an XML document in one language to an XML document in a second language. The translation of XML documents as taught by Worden would not enable a person of ordinary skill in the art to make Applicant's claimed subject matter which involves for example integrating data models for software components into a data wedge.

Therefore, the use of Worden to anticipate Applicant's invention is improper and Applicant requests that the anticipation rejection based on Worden be reverse by the Board of Appeals.

(C) 103(a) Rejection of Dependent Claims

The dependent claims are rejected as being obvious in view of Worden. As shown above for the 102(e) rejection, Worden fails to show or suggest all the elements of the independent claims and thus it follows that Worden fails to show or suggest all the elements of the claims that depend from the independent claims. Therefore, the Office has failed to establish a *prima facie* case of obviousness and the rejection is improper. The Board of Appeals is respectfully requested to reverse the rejection of these claims.

CONCLUSION

The Office has not established that the application failed to comply with the written description requirement and the Office further has not established that the cited reference either anticipated or made obvious Applicant's claimed invention. Accordingly, the Board of Appeals is respectfully requested to reverse the rejections of claims 17-33 and allow all pending claims.

Respectfully submitted,

Date: December 15, 2009

(Filed Electronically)

/Harden E. Stevens, III/

Harden E. Stevens, III
Reg. No. 55,649

NCR Corporation
1700 South Patterson Blvd.
Dayton, Ohio 45479

(803) 939-6505
(803) 939-5521 (fax)
Email: steve.stevens@ncr.com

(8) CLAIMS APPENDIX

1.-16. (Canceled)

17. A computer-implemented method of storing and translating data between a format of a first data model of a first software component and a format of a second data model of a second software component, the method comprising:

creating a first schema comprising the first data model of the first software component;

creating a second schema comprising the second data model of the second software component;

creating a third data model and a data storage in a data wedge by integrating the first schema and the second schema into the data wedge;

receiving a data element in the format of the first data model of the first software component, translating the data element from the format of the first data model of the first software component to the format of the third data model in the data wedge and storing the translated data element in the data storage, by the data wedge; and

retrieving the data element from the data storage and translating the data element into the format of the second data model of the second software component by the data wedge after receiving a request for the data element from the second software component.

18. The method of claim 17, further comprising:

triggering an event to notify the second software component of the availability of the data element received from the first software component and stored in the data wedge.

19. The method of claim 17, further comprising:

reading the data element translated into the format of the second data model by the second software component.

20. The method of claim 17, further comprising:

removing an obsolete data element from the first data model of the first software component and causing the data wedge to remove the translated data element from the third data model.

21. The method of claim 17, further comprising:

creating an instance of the data wedge.

22. The method of claim 17, wherein the first and second schemas further comprise a name of the data wedge.

23. The method of claim 17, wherein integrating the first schema into the data wedge includes setting default data elements and data values for the first data model of the first software component.

24. The method of claim 17, further comprising:

retrieving the data element from the data storage and translating the data element from the format of the third data model to the format of the first data model of the first software component by the data wedge after receiving a request for the data element from the first software component.

25. A computer system for translating data between a format of a first data model of a first software component and a format of a second data model of a second software component, the system comprising:

a processor; and

a memory coupled to said processor, the memory having stored therein data and sequences of instructions which, when executed by said processor, cause said processor to:

create a first schema comprising the first data model of the first software component;

create a second schema comprising the second data model of the second software component;

create a third data model and a data storage in a data wedge by integrating the first schema and the second schema into the data wedge;

receive a data element in the format of the first data model of the first software component, translate the data element from the format of the first data model of the first software component to a format of the third data model in the data wedge and store the translated data element in the data storage; and

retrieve the data element from the data storage and translate the data element into the format of the second data model of the second software component after receiving a request for the data element from the second software component.

26. The system of claim 25, further comprising instructions which, when executed by said processor, cause said processor to:

triggering an event to notify the second software component of the availability of the data element received from the first software component and stored in the data storage.

27. The system of claim 25, further comprising instructions which, when executed by said processor, cause said processor to:

remove an obsolete data element from the first data model of the first software component and remove the translated data element from the third data model.

28. The system of claim 25, further comprising instructions which, when executed by said processor, cause said processor to:

create an instance of the data wedge.

29. The system of claim 25, further comprising instructions which, when executed by said processor, cause said processor to:

retrieve the data element from the data storage and translate the data element from the format of the third data model to the format of the first data model of the first software component after receiving a request for the data element from the first software component.

30. The system of claim 25, wherein the first and second schemas further comprise a name of the data wedge.

31. The system of claim 25, wherein the instructions causing the processor to integrate the first schema into the data wedge include instructions causing the processor to set default data elements and data values for the first data model of the first software component.

32. A computer system for translating data between a format of a data model of a first software component and a format of a data model of a second software component, the system comprising:

a processor; and

a memory coupled to said processor,

wherein said processor is configured to execute a sequence of instructions contained in said memory, the instructions comprising a data wedge including a first schema of the first software component and a second schema of the second software component, the data wedge configured to translate a data element from the format of the data model of the first software component in accordance with the first schema to a data model of the data wedge and when a request is received from the second software component, translate the data element from the format of the data model of the data wedge to the format of the data model of the second software component in accordance with the second schema.

33. The system of claim 32, wherein said data wedge is further configured to trigger an event to notify the second software component of translated data element availability.

(9) EVIDENCE APPENDIX

None.

(10) RELATED PROCEEDINGS APPENDIX

None.